Symmetric Encryption: Construction from PRF

Notes by 6.5610 Staff

February 4, 2026

> ***Warning:*** This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

Outline

- Definition of symmetric encryption
- Definition of pseudorandom function (PRF)
- Construction of symmetric encryption from any PRF family
- PRF example

Defining an Encryption Scheme

In what follows, we define the notion of an encryption scheme. We start with the syntax.

Definition 1. A symmetric encryption scheme is associated with a key space $\{\mathcal{K}_\lambda\}_{\lambda\in\mathbb{N}}$, a message space $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ and a ciphertext space $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, and with two algorithms $(\mathsf{Enc}, \mathsf{Dec})$, where

$$\mathsf{Enc}_\lambda : \mathcal{K}_\lambda \times \mathcal{M}_\lambda \to \mathcal{C}_\lambda$$

and

$$\mathsf{Dec}_\lambda : \mathcal{K}_\lambda \times \mathcal{C}_\lambda \to \mathcal{M}_\lambda$$

Correctness: For every $\lambda \in \mathbb{N}$, every $m \in \mathcal{M}_\lambda$, and every $k \in \mathcal{K}_\lambda$,

$$\mathsf{Dec}_\lambda(k, \mathsf{Enc}_\lambda(k, m)) = m.$$

What about security? How do we define security?

If we only have security for random messages then one can encrypt a message m by choosing a random message $r \xleftarrow{R} \mathcal{M}$ and outputting $(\mathsf{Enc}(k, r), r \oplus m$. This can be thought of as a way of enhancing the security of an encryption scheme.

Defining Security

When defining security one needs to define what the adversarial goal is and what is its power. In the case of an encryption scheme, the adversarial goal is to break the encryption of *any* message. A weaker goal would be to break the encryption of random messages. This may be too weak since in practice we do not encrypt random messages.

Definition 2 (Take 1)**.** An encryption scheme is said to be secure if for every $\lambda \in \mathbb{N}$ and for every pair of messages $m_0, m_1 \in \mathcal{M}_\lambda$ it holds that

$$\mathsf{Enc}(k, m_0) \equiv \mathsf{Enc}(k, m_1)$$

where $k \xleftarrow{R} \mathcal{K}_\lambda$.

$\equiv$ means that the two distributions are equal, i.e.,

$$\forall\, c : \Pr_k\left[\mathsf{Enc}(k, m_0) = c\right] = \Pr_k\left[\mathsf{Enc}(k, m_1) = c\right].$$

Is Definition 2 strong enough? Here is a scheme that satisfies this definition, and is known as the one-time pad, invented by Vernam in 1917 (and used in WW1 and WW2):

$$\mathsf{Enc}(k, m) = k \oplus m$$

where $\mathcal{K}_\lambda = \mathcal{M}_\lambda = \{0,1\}^\lambda$.

$$\mathsf{Dec}(k, c) = k \oplus c$$

Indeed, the one-time pad satisfies Definition 2. This was proven formally by Shannon. However, if the adversary sees two ciphertexts c_1 and c_2 they can compute the xor of the underlying encrypted messages by computing $c_1 \oplus c_2$. In addition, if the adversary sees a single encryption c of a known message m then they can recover the secret key k by computing $k = c \oplus m$!

The important observation is that cirphertexts are functions of the secret key and therefore may leak information about the secret key. We need an encryption scheme that is secure even if the adversary sees many ciphertexts.

Definition 3 (Take 2)**.** An encryption scheme is said to be secure if for every $\lambda \in \mathbb{N}$, every $\ell \in \mathbb{N}$ and all messages $m_1, \ldots, m_\ell \in \mathcal{M}_\lambda$ and $m'_1, \ldots, m'_\ell \in \mathcal{M}_\lambda$ it holds that

$$\left(\mathsf{Enc}(k, m_1), \ldots, \mathsf{Enc}(k, m_\ell)\right) \equiv \left(\mathsf{Enc}(k, m'_1), \ldots, \mathsf{Enc}(k, m'_\ell)\right) \qquad (1)$$

where the randomness is over $k \leftarrow \mathcal{K}_\lambda$.

This seems like a sufficiently strong definition, but it is impossible to achieve! First, in order to have any hope of achieving this definition we must change the syntax and allow the encryption algorithm to be *randomized*, since if it is deterministic then the adversary can tell if the same message was encrypted twice.

Indeed, we allow the encryption algorithm to be randomized, and change the correctness condition in Definition 1, as follows:

Sometimes this completeness condition is weakened and the probability is allowed to be $1 - \mathrm{negl}(\lambda)$.

Correctness: For every $\lambda \in \mathbb{N}$ and for every $m \in \mathcal{M}_\lambda$ and every $k \in \mathcal{K}_\lambda$,

$$\Pr[\mathsf{Dec}_\lambda(k, \mathsf{Enc}_\lambda(k, m)) = m] = 1$$

where the probability is over the random coin tosses of Enc.

Unfortunately, even with a randomized encryption algorithm, achieving many-time security, as defined in Definition 3, is impossible! Intuitively, the reason is that each ciphertext contains some information about the secret key k, and eventually all the information about k will be contained in these ciphertexts.

One thing you will learn in this class is that cryptography has magical power to overcome impossibility results. In particular, the way we overcome this barrier is by relaxing the security requirement, and rather than requiring that the distributions in the left-hand-side and the right-hand-side of Equation 1 are the same, we require that they only look the same to polynomial time adversaries. Distributions that look the same to polynomial time adversaries are called *computationally indistinguishable*, and is denoted by $\approx$. We formalize this below.

Definition 4. A function $\mu : \mathbb{N} \to \mathbb{N}$ is said to be negligible if for every $c \in \mathbb{N}$ there exists $n_c \in \mathbb{N}$ such that for every $n > n_c$ it holds that $\mu(n) < n^{-c}$.

We denote by $a \leftarrow \mathcal{A}_\lambda$ if a is sampled from the distribution $\mathcal{A}_\lambda$. We denote by $k \xleftarrow{R} \mathcal{K}_\lambda$ is k is randomly chosen from the set $\mathcal{K}_\lambda$.

Definition 5. Let $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda \in \mathbb{N}}$ be two families of distributions. We say that $\mathcal{A}$ and $\mathcal{B}$ are computationally indistinguishable, denoted by $\mathcal{A} \approx \mathcal{B}$, if for every probabilistic polynomial time (PPT) distinguisher $\mathcal{D}$ there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$,

$$|\Pr[\mathcal{D}(a) = 1] - \Pr[\mathcal{D}(b) = 1]| \leq \mu(\lambda)$$

where $a \leftarrow \mathcal{A}_\lambda$ and $b \leftarrow \mathcal{B}_\lambda$.

λ is the security parameter. The larger the security parameter the more secure the scheme is, but also the less efficient it is.

Definition 6 (Take 3)**.** An encryption scheme is said to be secure if for every $\lambda \in \mathbb{N}$, every $\ell \in \mathbb{N}$ and all messages $m_1, \ldots, m_\ell \in \mathcal{M}_\lambda$ and $m'_1, \ldots, m'_\ell \in \mathcal{M}_\lambda$ it holds that

$$(\mathsf{Enc}(k, m_1), \ldots, \mathsf{Enc}(k, m_\ell)) \approx \left(\mathsf{Enc}(k, m'_1), \ldots, \mathsf{Enc}(k, m'_\ell)\right) \qquad (2)$$

This seems like a super strong security guarantee! However, the golden standard definition is even stronger! It also allows the adversary to see decryptions of ciphertexts of its choice. This is referred to as security against chosen ciphertext attacks (CCA-security).

The actual definition is a bit more complicated and allows the adversary to choose the messages adaptively based on previously seen ciphertexts. This is called security against *adaptively chosen plaintext attacks*.

Definition 7. An encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ is said to be secure against *adaptively chosen plaintext attacks* (CPA secure) if for every

PPT adversary $\mathcal{A}$ there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$, $\mathcal{A}$ wins in the following game with probability at most $\frac{1}{2} + \mu(\lambda)$:

- The challenger chooses a key $k \leftarrow \mathcal{K}_\lambda$.
- The adversary $\mathcal{A}$ given 1^λ chooses a message $m_i \in \mathcal{M}_\lambda$ and receives $c_i \leftarrow \mathsf{Enc}_\lambda(k, m_i)$.

  This step can be repeated polynomially many times.
- The adversary $\mathcal{A}$ chooses $m_0, m_1 \in \mathcal{M}_\lambda$.
- The challenger chooses a random bit $b \leftarrow \{0,1\}$, generates $c \leftarrow \mathsf{Enc}(k, m_b)$, and sends the ciphertext c to the adversary.
- The adversary given c outputs a bit b'.

We say that $\mathcal{A}$ wins if $b' = b$.

Constructing a CPA*-Secure Encryption Scheme*

The observation is that if the secret key was an infinitely long pad then we could use it to encrypt all our messages, each time using a fresh part of the pad. In other words, if the secret key was a perfectly random function $F : \{0,1\}^\lambda \to \{0,1\}$, then we could encrypt a message m as follows:

$$\mathsf{Enc}(F, m) = (r, F(r) \oplus m)$$

and

$$\mathsf{Dec}(F, (c_1, c_2)) = F(c_1) \oplus c_2.$$

As long as we encrypt significantly less than $2^{\lambda/2}$ messages we do not expect to see a collision (i.e., the same r used twice) and hence security follows from the one-time security of the one-time pad.

Here is the first magic of cryptography: We can construct efficiently computable functions that look like truly random ones! Such functions are called *pseudorandom functions*.

Definition 8 (Pseudorandom function)**.** A pseudorandom function family consists of a family of functions $\{F_\lambda\}_{\lambda \in \mathbb{N}}$, where for every $\lambda \in \mathbb{N}$, $F_\lambda : \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$, and for every PPT algorithm $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that for every $\lambda \in \mathbb{N}$,

$$|\Pr[\mathcal{A}^{F_\lambda(k,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{A}^{R_\lambda(\cdot)}(1^\lambda) = 1]| \leq \mu(\lambda)$$

where $k \xleftarrow{R} \mathcal{K}_\lambda$ and $R_\lambda : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$ is a truly random function; $\mathcal{A}$ has oracle access to $F_\lambda(k,\cdot)$ or $R_\lambda(\cdot)$, and can make arbitrary oracle calls to its function. These oracle calls $x_1, \ldots, x_t \in \mathcal{X}_\lambda$ can be adaptively chosen based on the values returned by the oracle thus far.

For concreteness, we can think of $\mathcal{X} = \{0,1\}^n$ and $\mathcal{Y} = \{0,1\}^m$.

In complexity theory, we model an efficient algorithm as polynomial time (or probabilistic polynomial time). We think of negligible as "practically never." $\mathcal{A}$ takes as input 1^λ since it is a PPT algorithm, and we allow it to run in time $\mathrm{poly}(\lambda)$. This is a notational hack used by theoreticians.

Using a PRF *to construct a* CPA*-secure encryption scheme*

Let $F = \{F_\lambda\}_{\lambda \in \mathbb{N}}$ be any PRF family where $F_\lambda : \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$. Suppose $\mathcal{Y} = \{0,1\}^{m(\lambda)}$. We use F to construct a symmetric encryption scheme where the key-space is $\mathcal{K}_\lambda$, the message space is $\{0,1\}^{m(\lambda)}$, and the ciphertext space is $\mathcal{X}_\lambda \times \{0,1\}^{m(\lambda)}$. Specifically,

$$\mathsf{Enc}_\lambda(k, m) = (r, m \oplus F(k, r))$$

where $r \xleftarrow{R} \mathcal{X}_\lambda$.

$$\mathsf{Dec}_\lambda(k, (r, c)) = F(k, r) \oplus c.$$

The CPA security of this scheme can be shown given the security of the underlying PRF, and we will concretely relate the success probabilities of the PPT adversaries in the next lecture.

PRF *Example*

Stream ciphers such as ChaCha20 used in TLS are PRFs.

$$F_k(i) = ChaCha20_k(nonce, i)$$

where i is a counter, and $F_k(i)$ is XOR'ed with the plaintext stream to produce a ciphertext stream. We will describe ChaCha20 in the next lecture.

PRF *for Integrity*

So far we saw how to use a PRF to encrypt. Suppose we wish to ensure authenticity of the ciphertext; namely, ensure that an adversary did not change the ciphertext, and that the ciphertext obtained was indeed the one sent by the sender. This can be done also using a PRF. I will "sign" a message c (the notation is due to the fact that I am thinking of this message as being a ciphertext) by appending to c the tag $F(k, c)$. This tag is called a *Message Authentication Code* (MAC). We will discuss this in more detail in the public-coin (where such a tag is referred to as a signature) setting later in this course.

References